User Datagram Program (UDP)

Courtesy: DC&N by B. A. Forouzan

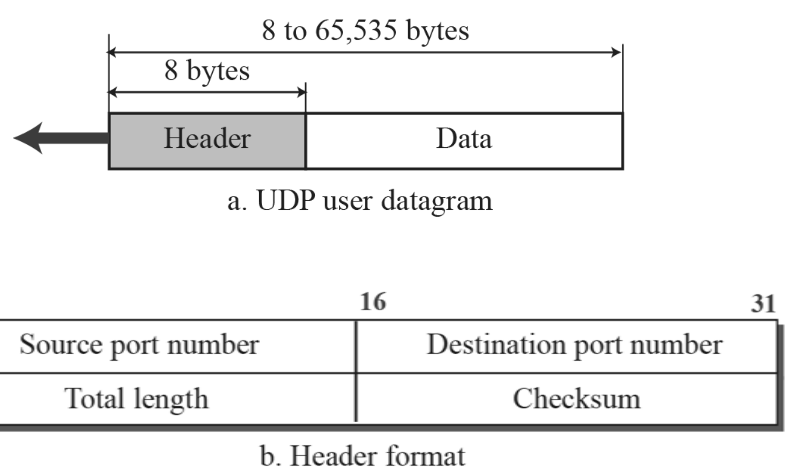
OBJECTIVES:

- To introduce UDP and show its relationship to other protocols in the TCP/IP protocol suite.
- To explain the format of a UDP packet and discuss the use of each field in the header.
- To discuss the services provided by the UDP such as process-to-process delivery, multiplexing/demultiplexing, and queuing.
- To show how to calculate the optional checksum and the sender the needs to add a pseudoheader to the packet when calculating the checksum.
- To discuss how some application programs can benefit from the simplicity of UDP.
- To briefly discuss the structure of the UDP package.

14-2 USER DATAGRAM

UDP packets, called user datagrams, have a fixed-size header of 8 bytes. Figure 14.2 shows the format of a user datagram.

Figure 14.2 *User datagram format*



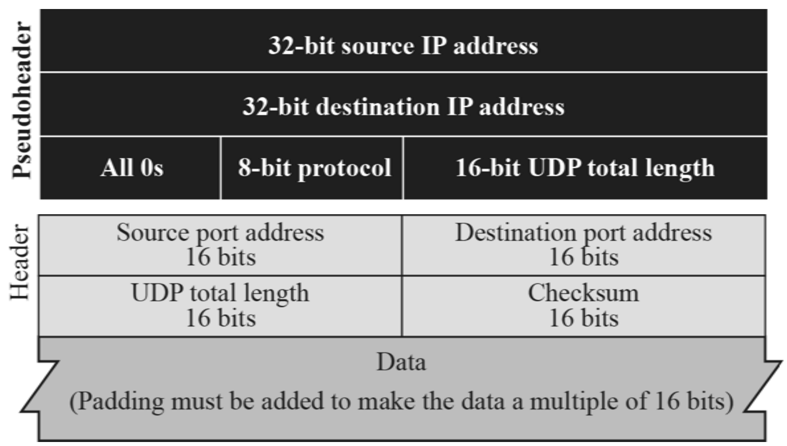
Topics Discussed in the Section

- ✓ Process-to-Process Communication
- ✓ Connectionless Service
- ✓ Flow Control
- ✓ Error Control
- ✓ Congestion Control
- ✓ Encapsulation and Decapsulation
- ✓ Queuing
- ✓ Multiplexing and Demultiplexing
- ✓ Comparison between UDP and Generic Simple Protocol

Table 14.1 *Well-known Ports used with UDP*

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Domain	Domain Name Service (DNS)
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

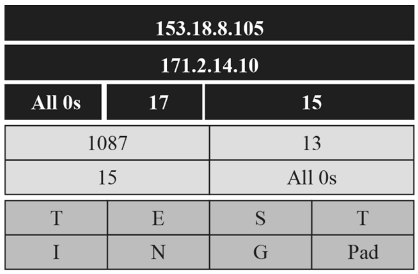
Figure 14.3 *Pseudoheader for checksum calculation*



Example 14.2

Figure 14.4 shows the checksum calculation for a very small user datagram with only 7 bytes of data. Because the number of bytes of data is odd, padding is added for checksum calculation. The pseudoheader as well as the padding will be dropped when the user datagram is delivered to IP

Figure 14.4 ***Checksum calculation for a simple UDP user datagram***



10011001 00010010 ⟶ 153.18
00001000 01101001 ⟶ 8.105
10101011 00000010 ⟶ 171.2
00001110 00001010 ⟶ 14.10
00000000 00010001 ⟶ 0 and 17
00000000 00001111 ⟶ 15
00000100 00111111 ⟶ 1087
00000000 00001101 ⟶ 13
00000000 00001111 ⟶ 15
00000000 00000000 ⟶ 0 (checksum)
01010100 01000101 ⟶ T and E
01010011 01010100 ⟶ S and T
01001001 01001110 ⟶ I and N
01000111 00000000 ⟶ G and 0 (padding)

10010110 11101011 ⟶ Sum
01101001 00010100 ⟶ Checksum

Example 14.3

What value is sent for the checksum in one of the following hypothetical situations?

a. The sender decides not to include the checksum.

b. The sender decides to include the checksum, but the value of the sum is all 1s.

c. The sender decides to include the checksum, but the value of the sum is all 0s.

Example 14.3 *Continued*

Solution

a. The value sent for the checksum field is all 0s to show that the checksum is not calculated.

b. When the sender complements the sum, the result is all 0s; the sender complements the result again before sending. The value sent for the checksum is all 1s. The second complement operation is needed to avoid confusion with the case in part a.

c. This situation never happens because it implies that the value of every term included in the calculation of the sum is all 0s, which is impossible; some fields in the pseudoheader have nonzero values.

Figure 14.5 *Encapsulation and decapsulation*

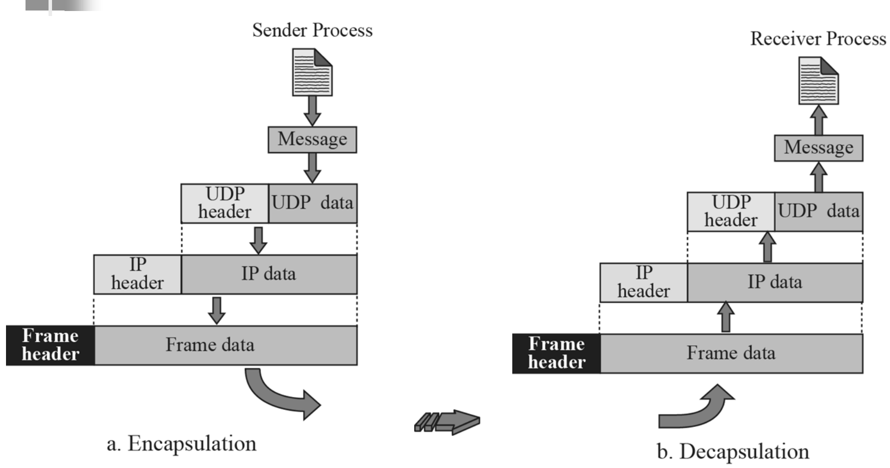
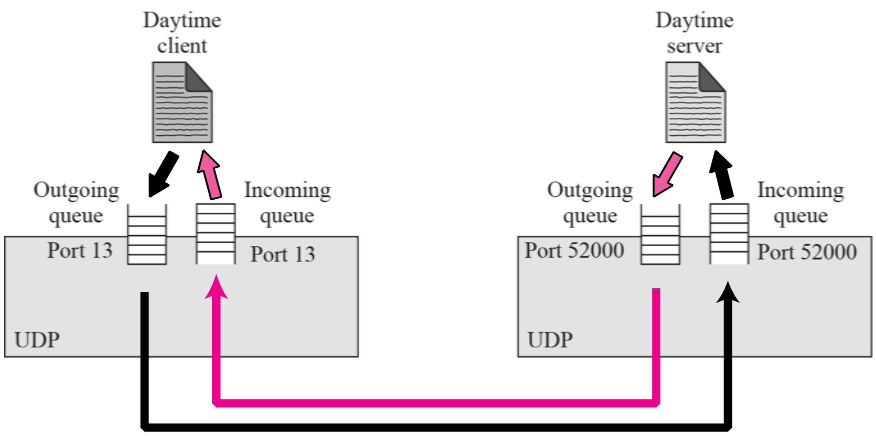
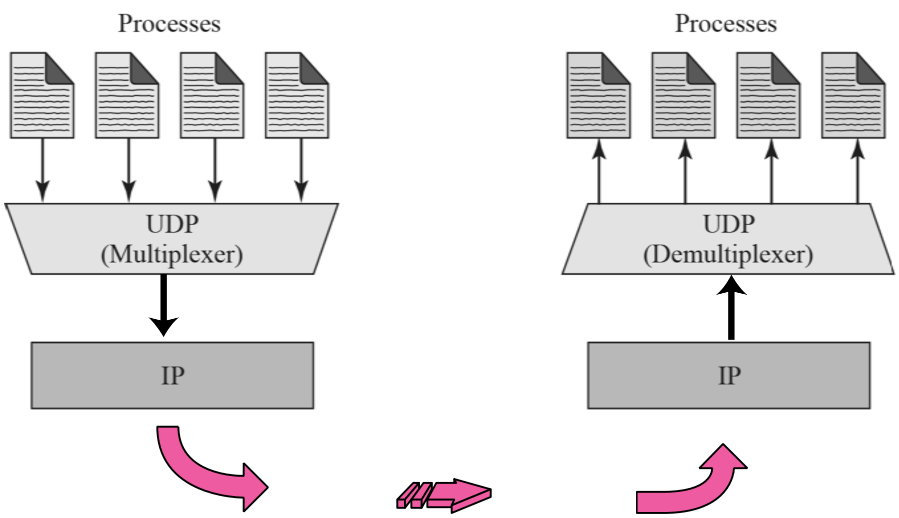


Figure 14.6 *Queues in UDP*



TCP/IP Protocol Suite 13

Figure 14.7 *Multiplexing and demultiplexing*



TCP/IP Protocol Suite 14

Note

UDP is an example of the connectionless simple protocol with the exception of an optional checksum added to packets for error detection.

14-4 UDP APPLICATION

Although UDP meets almost none of the criteria for a reliable transport-layer protocol, UDP is preferable for some applications. The reason is that some services may have some side effects that are either unacceptable or not preferable. An application designer needs sometimes to compromise to get the optimum.

Topics Discussed in the Section

- ✓ **UDP Features**
- ✓ **Typical Applications**

Example 14.4

A client-server application such as DNS uses the services of UDP because a client needs to send a short request to a server and to receive a quick response from it. The request and response can each fit in one user datagram. Since only one message is exchanged in each direction, the connectionless feature is not an issue; the client or server does not worry that messages are delivered out of order.

Example 14.5

A client-server application such as SMTP, which is used in electronic mail, cannot use the services of UDP because a user can send a long e-mail message, which may include multimedia (images, audio, or video). If the application uses UDP and the message does not fit in one single user datagram, the message must be split by the application into different user datagrams. Here the connectionless service may create problems. The user datagrams may arrive and be delivered to the receiver application out of order. The receiver application may not be able to reorder the pieces. This means the connectionless service has a disadvantage for an application program that sends long messages.

Example 14.6

Assume we are downloading a very large text file from the Internet. We definitely need to use a transport layer that provides reliable service. We don't want part of the file to be missing or corrupted when we open the file. The delay created between the delivery of the parts are not an overriding concern for us; we wait until the whole file is composed before looking at it. In this case, UDP is not a suitable transport layer.

Example 14.7

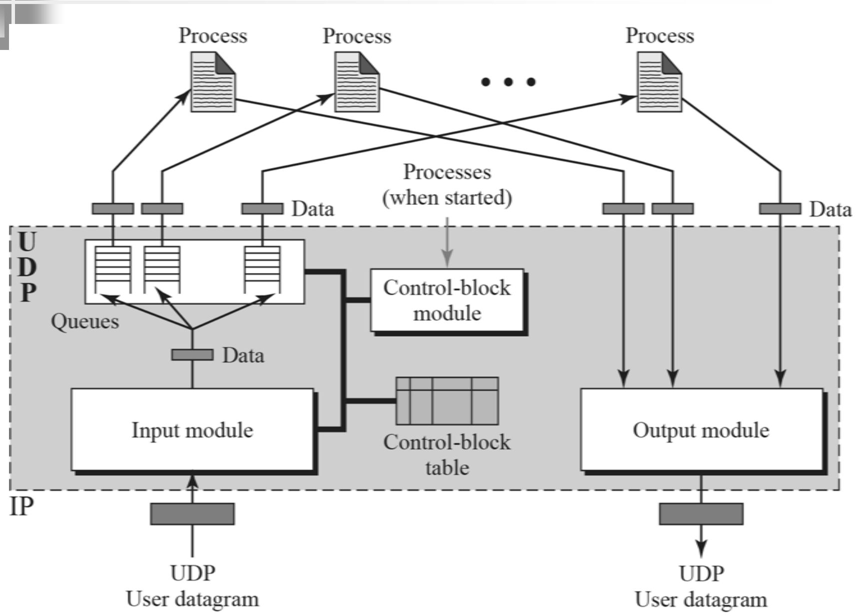
Assume we are watching a real-time stream video on our computer. Such a program is considered a long file; it is divided into many small parts and broadcast in real time. The parts of the message are sent one after another. If the transport layer is supposed to resend a corrupted or lost frame, the synchronizing of the whole transmission may be lost. The viewer suddenly sees a blank screen and needs to wait until the second transmission arrives. This is not tolerable. However, if each small part of the screen is sent using one single user datagram, the receiving UDP can easily ignore the corrupted or lost packet and deliver the rest to the application program. That part of the screen is blank for a very short period of the time, which most viewers do not even notice. However, video cannot be viewed out of order, so streaming audio, video, and voice applications that run over UDP must reorder or drop frames that are out of sequence.

14-5 UDP PACKAGE

To show how UDP handles the sending and receiving of UDP packets, we present a simple version of the UDP package.

We can say that the UDP package involves five components: a control-block table, input queues, a control-block module, an input module, and an output module.

Figure 14.8 *UDP design*



TCP/IP Protocol Suite 23

Example 14.8

The first activity is the arrival of a user datagram with destination port number 52,012. The input module searches for this port number and finds it. Queue number 38 has been assigned to this port, which means that the port has been previously used. The input module sends the data to queue 38. The control-block table does not change.

TCP/IP Protocol Suite 24

Example 14.9

After a few seconds, a process starts. It asks the operating system for a port number and is granted port number 52,014. Now the process sends its ID (4,978) and the port number to the control-block module to create an entry in the table. The module takes the first FREE entry and inserts the information received. The module does not allocate a queue at this moment because no user datagrams have arrived for this destination (see Table 14.6).

Table 14.6 *Control-Block Table after Example 14.9*

State	*Process ID*	*Port Number*	*Queue Number*
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	
IN-USE	4,978	52,014	
IN-USE	4,652	52,012	38
FREE			

Example 14.10

A user datagram now arrives for port 52,011. The input module checks the table and finds that no queue has been allocated for this destination since this is the first time a user datagram has arrived for this destination. The module creates a queue and gives it a number (43). See Table 14.7.

Table 14.7 *Control-Block Table after Example 14.10*

State	*Process ID*	*Port Number*	*Queue Number*
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	43
IN-USE	4,978	52,014	
IN-USE	4,652	52,012	38
FREE			

Example 14.11

After a few seconds, a user datagram arrives for port 52,222. The input module checks the table and cannot find an entry for this destination. The user datagram is dropped and a request is made to ICMP to send an unreachable port message to the source.

Answer

- The transport layer protocols used for real time multimedia, file transfer, DNS and email, respectively are:
  (A) TCP, UDP, UDP and TCP
  (B) UDP, TCP, TCP and UDP
  (C) UDP, TCP, UDP and TCP
  (D) TCP, UDP, TCP and UDP